TUGAS PEMROGRAMAN 3
PENGANTAR KECERDASAN BUATAN

Oleh :

Diva Annisa Febecca (1301204302)
Johanes Raphael Nandaputra (1301204243)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2022

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bayesian Learning adalah sebuah teknik pembelajaran yang menentukan parameter model dengan memaksimalkan probabilitas posterior dari parameter yang diberikan data pelatihan. Idenya adalah bahwa beberapa nilai parameter lebih konsisten dengan data yang diamati daripada yang lain. Dengan aturan *Bayes*, memaksimalkan probabilitas posterior sama dengan memaksimalkan apa yang disebut bukti model, yang didefinisikan sebagai probabilitas bersyarat dari data pelatihan yang diberikan parameter model. Bukti sering dapat didekati dengan beberapa formula tertutup atau dengan aturan pembaruan. Teknik *Bayesian* memungkinkan untuk menggunakan semua data untuk pelatihan alih-alih memesan pola untuk validasi silang parameter.

1.2 Permasalahan

Permasalahan tugas pemrograman ini adalah kami ditugaskan untuk membuat program pelatihan (*training*) dan pengujian (*testing*) dengan diberikan file **traintest.xlsx** yang terdiri dari dua sheet, yaitu train dan test yang dimana berisi dataset untuk problem klasifikasi biner (*Binary Classification*). Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari nomor baris data (*id*), fitur input ($x1$ sampai $x3$), dan output kelas (y). Fitur input terdiri dari nilai - nilai integer dalam range tertentu untuk setiap fitur. Sedangkan output kelas bernilai biner (0 atau 1). Pada tahap *training*, menghasilkan output berupa model sesuai metode yang kami gunakan, yaitu metode *Naive Bayes*. Sedangkan pada tahap *testing,* menghasilkan output berupa kelas (0 atau 1).

BAB II

PEMBAHASAN

2.1 Naive Bayes

Metode yang kami gunakan adalah *Naive Bayes*, yang mana *Naive Bayes* menggunakan teori probabilitas sebagai dasar teori. Salah satu kelebihan Naive Bayes Classifier adalah sederhana tetapi memiliki akurasi yang tinggi (Rish, 2001). Pada penelitian ini, atribut yang digunakan bertipe numerik, dimana untuk atribut dengan tipe numerik ada perlakuan khusus sebelum dimasukkan dalam *Naive Bayes*. Jika atribut bernilai numerik (kontinyu), biasanya diasumsikan memiliki distribusi *Gaussian* dengan dua parameter, yaitu Mean (μ), dan Standar Deviasi (σ) (Han dan Kamber, 2006). Untuk perhitungan atribut dengan data numerik, pertama dihitung rata-rata dari data *training* dengan rumus yang dinotasikan dengan persamaan

Rumus menghitung mean

$$\mu = \sum_{i=1}^{n} x_i$$

$$atau$$

$$\mu = \frac{x_1 + x_2 + x_3 \ldots + x_n}{n}$$

dimana :

μ : rata – rata hitung (mean)

xi : nilai sample ke -i

n : jumlah sampel

Setelah mendapatkan rata-rata dari masing masing atribut pada data training, selanjutnya dilakukan perhitungan standar deviasi dari masing-masing atribut dengan rumus yang dinotasikan dengan persamaan

Rumus simpangan baku

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

Dimana :

σ : standar deviasi

xi : nilai x ke -i

μ : rata-rata hitung

n : jumlah sampel

Langkah selanjutnya adalah menghitung nilai probabilitas untuk data uji yang mempunyai nilai numerik atau angka. Dikarenakan semua kelas menggunakan tipe data numerik, maka nilai probabilitas harus dihitung semua. Variabel hasil belum diketahui karena variabel tersebut merupakan hasil prediksi dari data yang dihitung (*x1, x2, x3*) . Untuk menghitung nilai distribusi *Gaussian* menggunakan rumus sebagai berikut

$$P\left(X_i = x_i | Y = y_j\right) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{-\frac{(x_i - \mu_{ij})^2}{2\sigma^2 ij}}$$

2.2 Data Training

Data *Training* merupakan data latih yang digunakan dalam melatih *Naive Bayes* dalam mengolah perhitungan dalam menghasilkan nilai keluaran yang diinginkan. Data *Training* yang digunakan adalah dataset yang telah diberikan dalam file **traintest.xlsx** pada sheet **train**.

2.3 Data Testing

Data *Testing* merupakan data untuk menguji dan mengetahui performa model. Data *Testing* yang digunakan adalah dataset yang telah diberikan dalam file **traintest.xlsx** pada sheet **test**.

BAB III

KODE PROGRAM

3.1 Library

Terdapat 4 library yang diimport untuk membantu proses training, testing, maupun membaca dan menyimpan file pada program ini.

```
!pip install XlsxWriter
import pandas as pd
import math
from openpyxl import load_workbook
import xlsxwriter
```

3.2 Membaca Data Latih/Uji

Untuk membaca data *traintest.xlsx*, kami menggunakan library *pandas* untuk membantu proses pembacaannya. File *traintest.xlsx* akan dibaca dan tiap *sheet*-nya akan dimasukkan ke dua variabel yang berbeda. Lalu, tiap data *train* maupun data *test*, masing-masing disimpan ke dalam list yang sudah dibuat, yaitu *dataTrain* dan *dataTest*.

```
def importData():
  # Buat list untuk tiap data
  dataTrain = []
  dataTest = []

  # Membaca file
  xls = pd.ExcelFile('traintest.xlsx')
  trainSheet = pd.read_excel(xls, 'train')
  testSheet = pd.read_excel(xls, 'test')

  # Memasukan data train ke dalam list dataTrain
  for train in range(len(trainSheet['id'])):
    dataTrain.append([])
    dataTrain[train].append(trainSheet['id'][train])
    dataTrain[train].append(trainSheet['x1'][train])
    dataTrain[train].append(trainSheet['x2'][train])
    dataTrain[train].append(trainSheet['x3'][train])
    dataTrain[train].append(trainSheet['y'][train])

  # Memasukan data test ke dalam list dataTest
  for test in range(len(testSheet['id'])):
    dataTest.append([])
    dataTest[test].append(testSheet['id'][test])
    dataTest[test].append(testSheet['x1'][test])
    dataTest[test].append(testSheet['x2'][test])
    dataTest[test].append(testSheet['x3'][test])
    dataTest[test].append(testSheet['y'][test])

  return dataTrain, dataTest
```

3.3 Pelatihan atau Training Model

Pada *training*, pertama kita mencari *mean* dari *x1, x2, x3 (input)* terlebih dahulu dengan data dipisah berdasarkan nilai *y (output)*. Setelah itu, kita mencari *standard deviation* dari data yang dipisah tersebut dan hasil *mean* dari masing-masing input. Hasil *mean* dan *standard deviation* masing-masing input akan disimpan ke dalam list yang sudah dibuat, yaitu *meanTrue, meanFalse, stdTrue, stdFalse,* dan me-*return* list tersebut.

```python
def mean(dataTrain):
  # Buat list mean berdasarkan output (1 = True, 0 = False)
  meanTrue = []
  meanFalse = []

  # inisiasi variabel
  totTrue = 0
  totFalse = 0

  totX1true = 0
  totX2true = 0
  totX3true = 0

  totX1false = 0
  totX2false = 0
  totX3false = 0

  # proses menghitung mean
  for i in range(len(dataTrain)):
    if (dataTrain[i][4] == 1):
      totX1true += dataTrain[i][1]
      totX2true += dataTrain[i][2]
      totX3true += dataTrain[i][3]
      totTrue += 1
    else:
      totX1false += dataTrain[i][1]
      totX2false += dataTrain[i][2]
      totX3false += dataTrain[i][3]
      totFalse += 1

  meanX1true = totX1true/totTrue
  meanX2true = totX2true/totTrue
  meanX3true = totX3true/totTrue
  meanX1false = totX1false/totFalse
  meanX2false = totX2false/totFalse
  meanX3false = totX3false/totFalse

  # menjadikan 2 angka belakang koma
  meanX1true = float("{:.2f}".format(meanX1true))
  meanX2true = float("{:.2f}".format(meanX2true))
  meanX3true = float("{:.2f}".format(meanX3true))
  meanX1false = float("{:.2f}".format(meanX1false))
  meanX2false = float("{:.2f}".format(meanX2false))
  meanX3false = float("{:.2f}".format(meanX3false))

  # memasukkan hasil tiap mean ke masing2 list
  meanTrue.extend([meanX1true, meanX2true, meanX3true])
  meanFalse.extend([meanX1false, meanX2false, meanX3false])

  return meanTrue, meanFalse
```

```python
def standardDeviation(dataTrain, meanTrue, meanFalse):
  # Buat list std berdasarkan output (1 = True, 0 = False)
  stdTrue = []
  stdFalse = []

  # inisiasi variabel
  totTrue = 0
  totFalse = 0

  totX1true = 0
  totX2true = 0
  totX3true = 0

  totX1false = 0
  totX2false = 0
  totX3false = 0

  # proses menghitung variance
  for i in range(len(dataTrain)):
    if (dataTrain[i][4] == 1):
      totX1true += ((dataTrain[i][1] - meanTrue[0]) ** 2)
      totX2true += ((dataTrain[i][2] - meanTrue[1]) ** 2)
      totX3true += ((dataTrain[i][3] - meanTrue[2]) ** 2)
      totTrue += 1
    else:
      totX1false += ((dataTrain[i][1] - meanFalse[0]) ** 2)
      totX2false += ((dataTrain[i][2] - meanFalse[1]) ** 2)
      totX3false += ((dataTrain[i][3] - meanFalse[2]) ** 2)
      totFalse += 1

  stdX1true = totX1true/(totTrue-1)
  stdX2true = totX2true/(totTrue-1)
  stdX3true = totX3true/(totTrue-1)
  stdX1true = math.sqrt(stdX1true)
  stdX2true = math.sqrt(stdX2true)
  stdX3true = math.sqrt(stdX3true)

  stdX1false = totX1false/(totFalse-1)
  stdX2false = totX2false/(totFalse-1)
  stdX3false = totX3false/(totFalse-1)
  stdX1false = math.sqrt(stdX1false)
  stdX2false = math.sqrt(stdX2false)
  stdX3false = math.sqrt(stdX3false)

  # menjadikan 2 angka belakang koma
  stdX1true = float("{:.2f}".format(stdX1true))
  stdX2true = float("{:.2f}".format(stdX2true))
  stdX3true = float("{:.2f}".format(stdX3true))
  stdX1false = float("{:.2f}".format(stdX1false))
  stdX2false = float("{:.2f}".format(stdX2false))
  stdX3false = float("{:.2f}".format(stdX3false))

  # memasukkan hasil tiap std ke masing2 list
  stdTrue.extend([stdX1true,stdX2true,stdX3true])
  stdFalse.extend([stdX1false,stdX2false,stdX3false])

  return stdTrue,stdFalse
```

3.4 Menyimpan Model Hasil Training

Pada fungsi ini, berguna untuk menyimpan file '**trainingModel.xlsx**' yang berguna untuk menyimpan data train yang dipisah berdasarkan nilai y (*output*) di *sheet* yang berbeda. Lalu, hasil dari *mean* dan *variance* tiap *input* akan dimasukkan sesuai dengan data yang diberikan.

```
def saveTrainingXlsx(dataTrain, meanTrue, meanFalse, varTrue, varFalse):
  # Membuat file trainingModel.xlsx
  workbook = xlsxwriter.Workbook('trainingModel.xlsx')
  worksheetTrue = workbook.add_worksheet("Recommended")
  worksheetFalse = workbook.add_worksheet("Not Recommended")

  # Mengisi header tiap kolom
  worksheetTrue.write(0,0,'id')
  worksheetTrue.write(0,1,'x1')
  worksheetTrue.write(0,2,'x2')
  worksheetTrue.write(0,3,'x3')
  worksheetTrue.write(0,4,'y')
  worksheetFalse.write(0,0,'id')
  worksheetFalse.write(0,1,'x1')
  worksheetFalse.write(0,2,'x2')
  worksheetFalse.write(0,3,'x3')
  worksheetFalse.write(0,4,'y')

  # Mengisi data tiap baris di sheet berdasarkan output
  idxTrue = 0
  idxFalse = 0
  for i in range(len(dataTrain)):
    if (dataTrain[i][4] == 1):
      worksheetTrue.write(idxTrue+1, 0, dataTrain[i][0])
      worksheetTrue.write(idxTrue+1, 1, dataTrain[i][1])
      worksheetTrue.write(idxTrue+1, 2, dataTrain[i][2])
      worksheetTrue.write(idxTrue+1, 3, dataTrain[i][3])
      worksheetTrue.write(idxTrue+1, 4, dataTrain[i][4])
      idxTrue += 1
    else:
      worksheetFalse.write(idxFalse+1, 0, dataTrain[i][0])
      worksheetFalse.write(idxFalse+1, 1, dataTrain[i][1])
      worksheetFalse.write(idxFalse+1, 2, dataTrain[i][2])
      worksheetFalse.write(idxFalse+1, 3, dataTrain[i][3])
      worksheetFalse.write(idxFalse+1, 4, dataTrain[i][4])
      idxFalse += 1

  # Menyimpan model hasil training

  # Starting point
  startTrue = idxTrue + 1
  startFalse = idxFalse + 1

  # Penambahan judul di baris kosong paling bawah tiap sheet
  worksheetTrue.write(startTrue, 0, "MEAN")
  worksheetTrue.write(startTrue+1, 0, "STD")
  worksheetFalse.write(startFalse, 0, "MEAN")
  worksheetFalse.write(startFalse+1, 0, "STD")

  # Mengisi data mean di tiap sheet
  worksheetTrue.write(startTrue, 1, meanTrue[0])
  worksheetTrue.write(startTrue, 2, meanTrue[1])
  worksheetTrue.write(startTrue, 3, meanTrue[2])
  worksheetFalse.write(startFalse, 1, meanFalse[0])
  worksheetFalse.write(startFalse, 2, meanFalse[1])
  worksheetFalse.write(startFalse, 3, meanFalse[2])

  # Mengisi data variance di tiap sheet
  worksheetTrue.write(startTrue + 1, 1, varTrue[0])
  worksheetTrue.write(startTrue + 1, 2, varTrue[1])
  worksheetTrue.write(startTrue + 1, 3, varTrue[2])
  worksheetFalse.write(startFalse + 1, 1, varFalse[0])
  worksheetFalse.write(startFalse + 1, 2, varFalse[1])
  worksheetFalse.write(startFalse + 1, 3, varFalse[2])

  # Menutup dan simpan file
  workbook.close()
```

3.5 Pengujian atau Testing Model

Dengan menggunakan *naive bayes*, akan dicari nilai tiap data test dari dua output yang berbeda dengan menggunakan rumus probabilitas *Gaussian*. Setelah itu, hasil tiap *Gaussian* akan disimpan ke dalam *list* yang telah dibuat dan me-*return*, yang nantinya akan dievaluasi.

```
def naiveBayes(dataTrain, dataTest, meanTrue, meanFalse, varTrue, varFalse):
  # inisiasi nilai pi dan e
  pi = 3.1416
  e = 2.7183

  # buat list untuk menampung
  recommendedlist = []
  notRecommendedList = []

  # inisiasi variabel
  totTrue = 0
  totFalse = 0

  # cari total data
  for i in range(len(dataTrain)):
    if (dataTrain[i][4] == 1):
      totTrue += 1
    else:
      totFalse += 1

  # hitung nilai dengan rumus Gaussian
  for data in range(len(dataTest)):
    recommended = (totTrue / len(dataTrain)) * (e ** -((((dataTest[data][1] - meanTrue[0]) ** 2) / (2 * (varTrue[0] ** 2)))) / (varTrue[0] * math.sqrt(2 * pi))) * (e **
    notRecommended = (totFalse / len(dataTrain)) * (e ** -((((dataTest[data][1] - meanFalse[0]) ** 2) / (2 * (varFalse[0] ** 2)))) / (varFalse[0] * math.sqrt(2 * pi)))

    recommendedlist.append(recommended)
    notRecommendedList.append(notRecommended)

  return recommendedlist, notRecommendedList
```

3.6 Evaluasi Model

Pada evaluasi model, hasil dari perhitungan *Gaussian* tiap *id* berdasarkan *output* yang telah dilakukan akan dibandingkan satu sama lain. Nilai yang lebih besar berarti jenis *output*-nya yang akan dimasukkan ke dalam *output* data *testing*.

```
def modelEvaluation(dataTest, recommended, notRecommended):
  for data in range(len(dataTest)):
  # bandingkan kedua nilai tersebut dan masukkan ke dataTest
    if recommended > notRecommended:
      dataTest[data][4] = 1
    else:
      dataTest[data][4] = 0

  return dataTest
```

3.7 Menyimpan Output ke File

Pada hasil *testing*, data *output*-nya akan disimpan kembali ke dalam file '**traintest.xlsx**' dengan tanda '?' pada awalnya akan tergantikan dengan hasil *testing* yang dilakukan.

```
def outputXlsx(dataTest):
  book = load_workbook("traintest.xlsx")
  testSheet = book['test']

  # memasukkan hasil output dari testing ke dalam file 'traintest.xlsx'
  for i in range(len(dataTest)):
    testSheet["E"+str(i+2)].value = dataTest[i][4]
  book.save("traintest.xlsx")
```

3.7 Main Program

Berikut program utama yang dibuat untuk melakukan prosesnya.

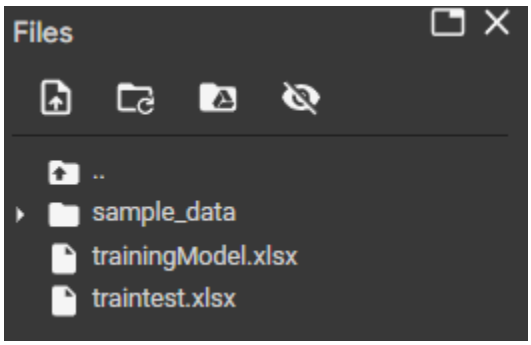
```
# MAIN PROGRAM

# Training
print("Tekan ENTER untuk memulai TRAINING!")
input()

print("===== MEMULAI TRAINING SESSION =====")
dataTrain, dataTest = importData()
meanTrue, meanFalse = mean(dataTrain)
stdTrue, stdFalse = standardDeviation(dataTrain, meanTrue, meanFalse)
saveTrainingXlsx(dataTrain, meanTrue, meanFalse, stdTrue, stdFalse)

print()
print("TRAINING SELESAI! Silakan cek file 'trainingModel.xlsx'!")
print()

print("=======================================================")

# Testing
print()
print("Tekan ENTER untuk memulai TESTING!")
input()

print("===== MEMULAI TESTING SESSION =====")
recommendedList, notRecommendedList = naiveBayes(dataTrain, dataTest, meanTrue, meanFalse, stdTrue, stdFalse)
dataTest = modelEvaluation(dataTest, recommendedList, notRecommendedList)
outputXlsx(dataTest)

print()
print("TRAINING selesai. Data telah diubah, silakan cek file!")
```

BAB IV

HASIL PERCOBAAN DAN KESIMPULAN

Setelah program utama berjalan, bagian *Training* akan menghasilkan file baru yang bernama 'trainingModel.xlsx', sedangkan file 'traintest.xlsx' yang dimasukkan paling awal, output *data testing* akan diperbaharui berdasarkan hasil *Naive Bayes.*

```
Tekan ENTER untuk memulai TRAINING!

===== MEMULAI TRAINING SESSION =====

TRAINING SELESAI! Silakan cek file 'trainingModel.xlsx'!

======================================================

Tekan ENTER untuk memulai TESTING!

===== MEMULAI TESTING SESSION =====

TRAINING selesai. Data telah diubah, silakan cek file!
```



Berikut isi file 'trainingModel.xslx' dan file 'traintest.xlsx' yang sudah ter-*update*:

215	290	64	61	0	1
216	291	50	61	6	1
217	292	59	64	1	1
218	293	65	67	0	1
219	296	54	59	7	1
220	MEAN	51,94	62,93	2,84	
221	STD	11,11	3,22	5,95	
222					

Recommended | Not Recommended

75	279	47	62	0	0
76	284	44	64	6	0
77	285	34	59	0	0
78	294	53	65	12	0
79	295	57	64	1	0
80	MEAN	53,94	62,76	7,67	
81	STD	10,2	3,28	9,3	
82					

Recommended | Not Recommended

Src = trainingModel.xlsx

id	x1	x2	x3	y
297	43	59	2	1
298	67	66	0	1
299	58	60	3	1
300	49	63	3	1
301	45	60	0	1
302	54	58	1	1
303	56	66	3	1
304	42	69	1	1
305	50	59	2	1
306	59	60	0	1

Src = traintest.xslx (updated)

Jadi, dari hasil *training* dan *testing* yang telah dilakukan pada file 'traintest.xlsx' dengan tipe *continuous data* ini dapat disimpulkan bahwa metode *naive bayes* dapat memperoleh hasil yang optimal dengan *training* sangat cepat dan mudah dan *testing* yang mudah.

DAFTAR PUSTAKA

Han, J., and Kamber, M., 2006, Data Mining : Concepts and Techniques, Second Edition, Morgan Kauffmann, San Francisco.

Rish, I., 2001, An Empirical Study Of The Naive Bayes Classifier, IBM Research Report, Thomas J. Watson Research Center, Yorktown Heights, New York.

http://eprints.binadarma.ac.id/3368/1/SPK%20Metode%20Bayes-%20MATRIK%20edisi%20april%202017-%20diana.pdf

LAMPIRAN

Nama	NIM	Peran
Johanes Raphael Nandaputra	1301204243	Coding, Laporan
Diva Annisa Febecca	1301204302	Coding, Laporan

Link Source Code :
https://colab.research.google.com/drive/14PpBCMGecWngBnTf_hyAMusOaLn_-UuE?usp=sharing